CASE STUDY

twenty four

Poker Simulation

case study

OVERVIEW

CS24.1 *Application Overview and Model Development*

This application calculates the chance of winning a game of poker given the hand that the user is dealt and the number of players in the game. The user can either have a random hand dealt or specify his or her hand. The total number of players in a game of poker can range from 2 to 5, and the ranking of poker hands (from lowest to highest) is as follows: High Card, One Pair, Two Pairs, Three Of a Kind, Straight, Flush, Full House, Four Of a Kind, and Straight Flush.

CS24.1.1 Model Definition and Assumptions

We assume the players are playing with one deck of 52 cards; there are four suits with 13 cards in each suit. If the user chooses to be dealt a random hand, then the entire deck is shuffled and each player is dealt five cards. If however, the user selects his or her own hand, these cards are moved to the front of the deck first and then the remainder of the deck is shuffled to deal the other players.

To shuffle the deck, we manipulate an array of card values; this array is size 52 by 2, in which the two dimensions store the suit and the card value separately. We set a counter variable *i* equal to 1 (if the user wants a random hand) or 6 (if the user has already specified his or her cards) and choose a random number *j* between *i* and 52. We switch the cards in position *j* with position *i* and increase *i* by one.

```
For i = 1 (or 6) to 52
        j = Int((52-i + 1)*Rnd() + i)
        Switch Card(i) and Card(j)
Next i
```

Once the deck is shuffled, we deal the user first and then each of the other players. Each player is dealt five cards from the front of the deck, i.e. from the beginning of the card array. That is, the user will be dealt cards in the deck position 1 to 5, player 2 may have cards in position 6 to 10, and so on. We store each player's hand in an array of size five by two (the second dimension is used to store the card number and card suit separately). The algorithm for dealing the remaining players after the user has been dealt is as follows:

```
For p = 2 to NumberPlayers
        For i = 1 to 5
                PlayerHand(p, i) = Card(i + (p-1)*5)
        Next i
Next p
```

After shuffling and dealing the cards, we sort each player's hand. We sort the cards in ascending order, leaving the high card as the last card in the player's card array. To perform the sort, we use a simple bubble sort algorithm. (See Chapter 17 for details on this sorting algorithm.)

To determine who has won the game, we compare the status of each player's hand. There are nine possible status values corresponding to the nine possible poker hands. For each run, we keep track of whether or not the user has the highest ranking hand. After the simulation is finished, we find the probability of the user winning with the dealt hand and

the given number of players by dividing the number of runs in which the user won by the total number of runs performed.

ProbUserWins = NumWins / NumRuns

Note that this probability of winning is calculated after all the runs are completed. That is, the last game (run) displayed to the user is not the situation for which the probability of winning is calculated. For example, if the last hand played gives the user a pair of sixes and his opponent a pair of aces, that particular game gives the user a zero percent chance of winning. However, the probability calculated considers how many times the user won with a pair of sixes for various opponent hands. Therefore, the probability displayed may be 0.61 percent chance of winning with a pair of sixes, although in the last game (shown in the final display) the user may have lost to a pair of aces.

CS24.1.2 Input

The input for this application is the following:

- Number of players
- Number of runs (number of games played)
- How the user's hand is dealt (random or specified)
- If the user's hand is specified, the five cards (suit and value) selected

CS24.1.3 Output

The output for this application is the following:

- The probability of winning with the dealt hand and the given number of players

CS24.2 *Worksheets*

This application requires three worksheets: the welcome sheet, the simulation sheet, and a hidden image sheet. The welcome sheet contains the title, the description of the application, and the "Start" button. (See Figure CS24.1). The "Start" button shows the user some forms and then takes him or her to the simulation sheet.

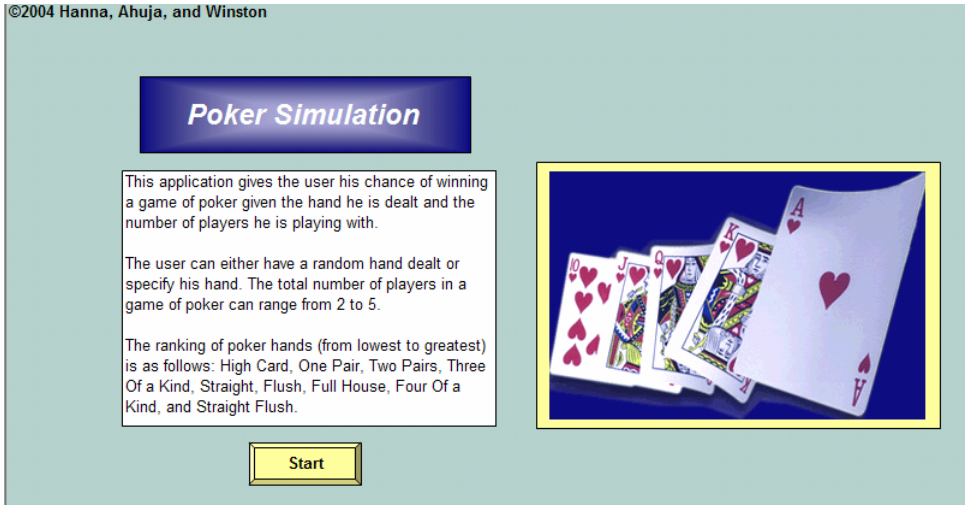

Figure CS24.1 The welcome sheet.

The simulation sheet is the main sheet of the application. (See Figure CS24.2.) It displays the user's hand, the number of players in the game, and the user's chance of winning for this given hand and number of players. It also shows the other players' hands for the last run of the simulation. There are three buttons on this sheet; two are used for the re-solve options, and the "End" button takes the user back to the welcome sheet.

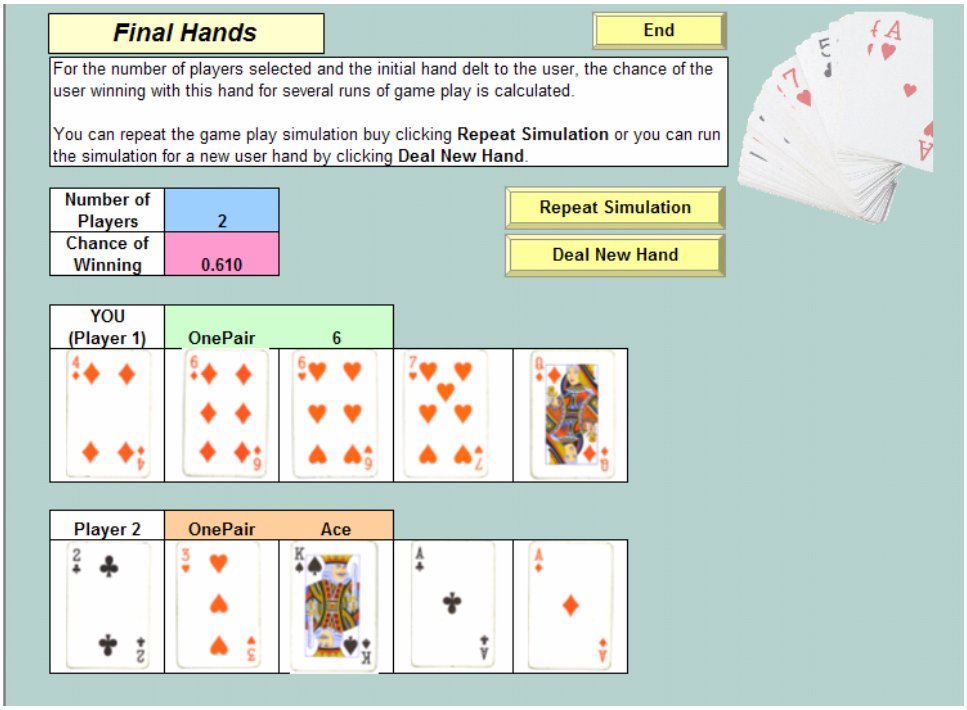

Figure CS24.2 The simulation sheet.

The third sheet, which is hidden, stores the card images by suit in increasing order of their value. (See Figure CS24.3.) This organization makes the task of copying and pasting players' hands easier; we will discuss this more in the "Procedures" section.

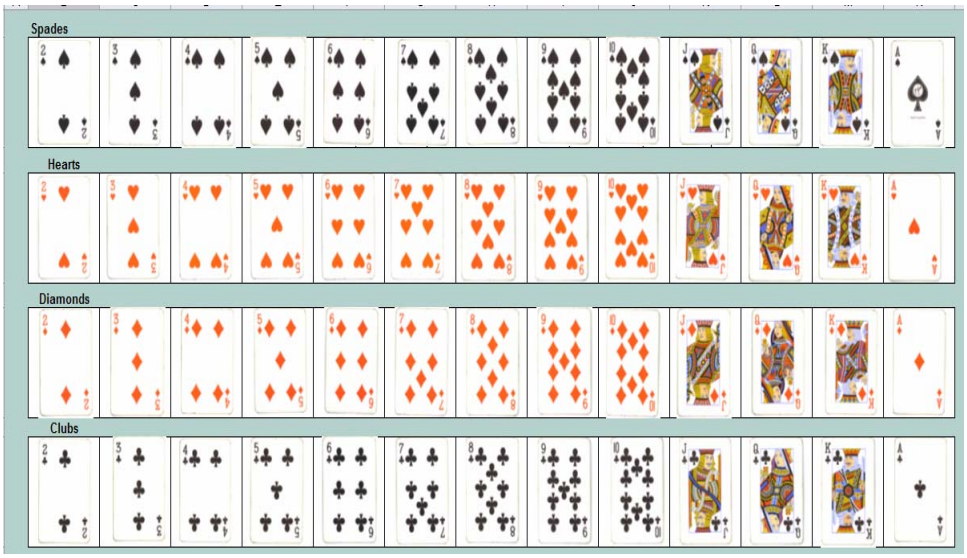

Figure CS24.3 The hidden image sheet.

Summary		
	Welcome sheet	Includes the application description and the "Start" button.
	Simulation sheet	The main sheet of the application, it displays the chance of winning, the number of players, the user's hand, and the other players' hands.
	Hidden image sheet	Stores the card images for each suit.

CS24.3 *User Interface*

For this application's user interface, we use navigational and functional buttons as well as two user forms. On the welcome sheet, the "Start" button brings the user to the simulation sheet. On the simulation sheet, the "End" button brings the user back to the welcome sheet. The "Repeat Simulation" and "Deal New Hand" buttons are functional buttons for the re-solve options; we will discuss them in more detail in Section CS4.5.

The application contains two user forms: the input form and the hand selection form. The former asks the user to specify the number of players that he or she wants to play with, the number of runs to perform in the simulation, and how to deal his or her hand. (See Figure CS24.4.) The user must play with at least one other player and at most four others. A spin button and text box collect this value. The user's hand can be dealt randomly from a shuffled deck, or he or she can specify a specific hand. We use options buttons for this input.

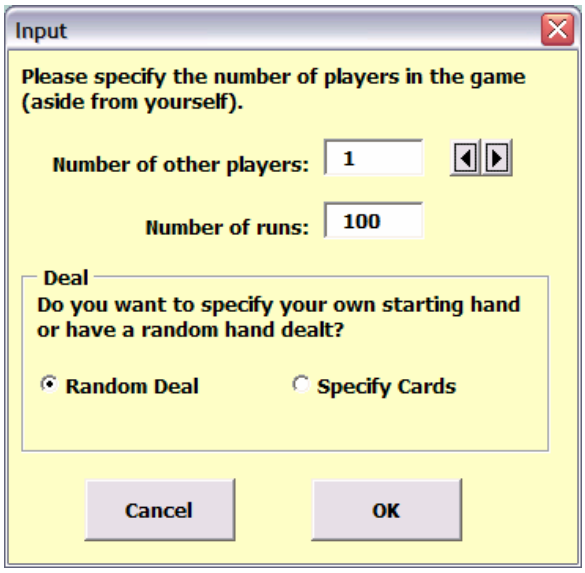

Figure CS24.4 The input form.

If the user opts to specify his or her own hand, then the hand selection form is displayed, (See Figure CS24.5.) This form uses combo boxes for the suit and the value of each of the five cards in the user's hand.

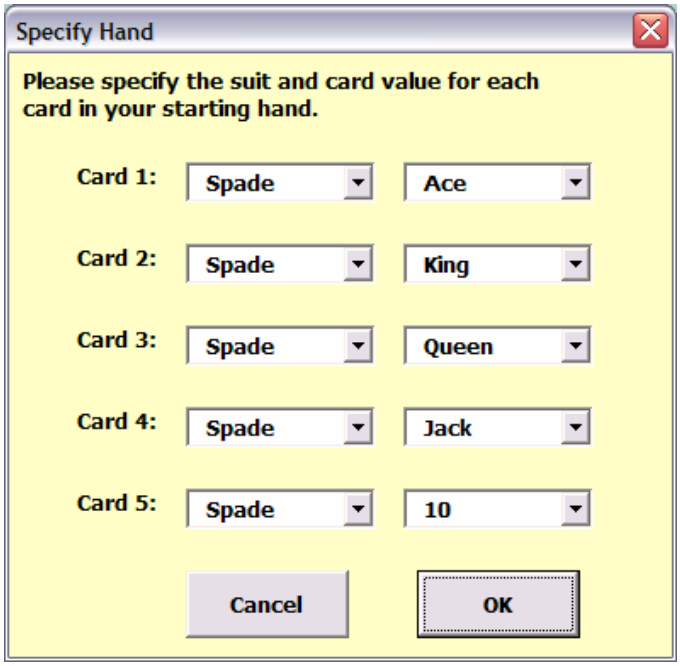

Figure CS24.5 The hand selection form.

Summary

Navigational buttons	"Start" on the welcome sheet; "End" on the simulation sheet.
Functional buttons	"Repeat Simulation" and "Deal New Hand" on the simulation sheet.
Input form	The number of players, the number of runs, how the user hand is dealt (random or specified).
Hand selection form	Where the suit and the value of each card are selected for the user's hand.

CS24.4 *Procedures*

We will now outline the procedures for this application, beginning with the initial sub procedures and the variable definitions. (See Figure CS24.6.) The "Main" procedure, which is called from the "Start" button, calls the "ClearPrev" procedure and shows the user the input form. It then formats the simulation sheet by hiding the display rows for the players not playing. Then, it calls the "RunSim" procedure and takes the user to the simulation sheet.

The input form procedures record the number of players and the number of runs assigned by the user. (See Figure CS24.7.) It also sets a Boolean variable value to record the user's option about whether or not to receive a random hand or to specify a certain hand. If the user opts to pick his or her own hand, then the hand selection form is also displayed.

The selection hand form uses an array of controls for the combo boxes for the card suits and another array for the combo boxes for the card values. (See Figure CS24.8.) It initializes these variables in the initialization procedure. (See Figure CS24.9.) After recording the suit and the value for each card selected by the user, we perform some error checking to ensure that the same card was not selected more than once. We do so with

an array to store which cards were selected. (Is there an alternate way to perform this error check?)

```
Option Explicit
Option Base 1

Public ws As Worksheet, NPlayers As Integer, i As Integer, j As Integer, k As Integer, _
Card(52, 2) As Variant, TempSuit As String, TempCard As Integer, _
PlayerHands(5, 5, 2) As Variant, RandHand As Boolean, CountVal(2 To 14) As Integer, _
CountSuit(4) As Integer, Status As String, HighCard As Integer, Straight As Boolean, _
Taken(52) As Boolean, DealStart As Integer, ChanceVal As Double, StatusVal(9) As String, _
Prob(9) As Double, WinStatus As Integer, Winner As String, WinCard As Integer, _
CountWin As Integer, runs As Integer, NRuns As Integer, p As Integer, _
Suits(5) As Control, Cards(5) As Control, NoChange As Boolean


Sub Main()  'called from Start button
    Call ClearPrev
    frmInput.Show

    'ensure rows for players playing are shown
    For i = 1 To NPlayers
        Range(Range("Player" & i), Range("Player" & i).Offset(1, 0)).Rows.Hidden = False
    Next i
    'hide rows for players not playing
    For i = NPlayers + 1 To 5
        Range(Range("Player" & i), Range("Player" & i).Offset(1, 0)).Rows.Hidden = True
    Next i

    Call Initialize
    Call RunSim

    Application.ScreenUpdating = True
    Worksheets("Final Hand").Visible = True
    Worksheets("Welcome").Visible = False
End Sub
```

Figure CS24.6 The variable declarations and the “Main” procedure.

```
Private Sub cmdCancel_Click()
    Unload Me
    Worksheets("Welcome").Visible = True
    Worksheets("Final Hand").Visible = False
    End
End Sub

Private Sub cmdOK_Click()
    NPlayers = txtNumPlayers.Value + 1
    Range("NumPlayers").Value = NPlayers
    NRuns = txtRuns.Value

    If optRandDeal Then
        RandHand = True
    ElseIf optSpecCards Then
        RandHand = False
        frmHandSelect.Show
    End If

    Unload Me
End Sub

Private Sub spnNumPlayers_Change()
    txtNumPlayers.Value = spnNumPlayers.Value
End Sub
```

Figure CS24.7 The input form procedures.

```
Private Sub cmdCancel_Click()
    Unload Me
    Worksheets("Welcome").Visible = True
    Worksheets("Final Hand").Visible = False
    End
End Sub

Private Sub cmdOK_Click()
    For i = 1 To 5
        PlayerHands(1, i, 1) = Suits(i).Value

        Select Case Cards(i).Value
            Case "Jack"
                PlayerHands(1, i, 2) = 11
            Case "Queen"
                PlayerHands(1, i, 2) = 12
            Case "King"
                PlayerHands(1, i, 2) = 13
            Case "Ace"
                PlayerHands(1, i, 2) = 14
            Case Else
                PlayerHands(1, i, 2) = CInt(Cards(i).Value)
        End Select
    Next i

    'must check that no cards are repeated
    For i = 1 To 52
        Taken(i) = False
    Next i
```

Figure CS24.8 The first part of the hand selection form procedures.

```
    For i = 1 To 5
        Select Case PlayerHands(1, i, 1)
            Case "Spade"
                j = 0
            Case "Heart"
                j = 13
            Case "Diamond"
                j = 26
            Case "Club"
                j = 39
        End Select

        k = PlayerHands(1, i, 2) - 1

        If Taken(j + k) = True Then
            MsgBox "You can not have repeated cards"
            Exit Sub
        Else
            Taken(j + k) = True
        End If
    Next i

    Unload Me
End Sub

Private Sub UserForm_Initialize()
    Set Suits(1) = cboSuit1
    Set Suits(2) = cboSuit2
    Set Suits(3) = cboSuit3
    Set Suits(4) = cboSuit4
    Set Suits(5) = cboSuit5

    Set Cards(1) = cboCard1
    Set Cards(2) = cboCard2
    Set Cards(3) = cboCard3
    Set Cards(4) = cboCard4
    Set Cards(5) = cboCard5

    For i = 1 To 5
        Suits(i).RowSource = "Suits"
        Suits(i).Value = Suits(i).List(0)

        Cards(i).RowSource = "Cards"
        Cards(i).Value = Cards(i).List(13 - i)
    Next i
End Sub
```

Figure CS24.9 The second part of the hand selection form procedures.

The “ClearPrev” procedure clears the status and card value cells from the simulation sheet for all the players. (See Figure CS24.10.) It also deletes all the card images from the simulation sheet.

```
Sub ClearPrev()
    'clear player status cells and card value cells
    For i = 1 To 5
        Range(Range("Player" & i).Offset(0, 1), Range("Player" & i).Offset(0, 2)).ClearContents
        Range(Range("Player" & i).Offset(1, 0), Range("Player" & i).Offset(1, 0).End(xlToRight)).ClearContents
    Next i

    'clear all card images on simulation sheet
    Worksheets("Final Hand").Activate
    Dim ImgObj As Object
    For Each ImgObj In ActiveSheet.DrawingObjects
        If ImgObj.Name <> "EndButton" And ImgObj.Name <> "DealButton" _
            And ImgObj.Name <> "CardImg" And ImgObj.Name <> "Title" And ImgObj.Name <> "Descrip" Then
            ImgObj.Delete
        End If
    Next

    NoChange = False
End Sub
```

Figure CS24.10 The “ClearPrev” procedure.

```
Sub RunSim()
    'run simulation
    CountWin = 0
    Application.ScreenUpdating = False

    'check if simulation is being repeated for same user hand
    If NoChange = False Then
        Call DealUser   'otherwise deal user hand
    Else
        'clear card images for other players
        Worksheets("Final Hand").Activate
        Dim ImgObj As Object
        For Each ImgObj In ActiveSheet.DrawingObjects
            If ImgObj.Name <> "EndButton" And ImgObj.Name <> "DealButton" _
                And ImgObj.Name <> "CardImg" And ImgObj.Name <> "Title" And ImgObj.Name <> "Descrip" Then
                ImgObj.Delete
            End If
        Next
        're-paste images of user's hand
        p = 1
        For i = 1 To 5
            Call GetImage(i)
            ActiveSheet.Paste Destination:=Range("Player1").Offset(1, i - 1)
        Next i
    End If

    'deal other players multiple times and check if user wins with dealt hand
    For runs = 1 To NRuns
        DealStart = 6
        Call Shuffle
        Call DealOthers
        Call CalcChance
    Next runs

    ChanceVal = CountWin / NRuns
    Range("Chance") = ChanceVal
End Sub
```

Figure CS24.11 The “RunSim” procedure.

The “RunSim” procedure, the main procedure of the application, performs the simulation (see Figure CS24.11.) It begins by calling an “Initialize” procedure, which initializes the array for all the card values and suits (see Figure CS24.12) Next, it calls the “DealUser” procedure, which deals the user’s hand (see Figure CS24.13). (Note that this procedure is

only called if the application is not being re-solved.) It then performs the simulation by calling the “Shuffle,” “DealOthers,” and “CalcChance” procedures for the given number of runs. Finally, the winning probability is calculated and displayed to the user.

```
Sub Initialize()
    'give suite and number values to all cards in deck
    For i = 1 To 52
        Select Case i
            Case 1 To 13
                Card(i, 1) = "Spade"
                Card(i, 2) = i + 1
            Case 14 To 26
                Card(i, 1) = "Heart"
                Card(i, 2) = i - 12
            Case 27 To 39
                Card(i, 1) = "Diamond"
                Card(i, 2) = i - 25
            Case 40 To 52
                Card(i, 1) = "Club"
                Card(i, 2) = i - 38
        End Select
    Next i
End Sub
```

Figure CS24.12 The “Initialize” procedure.

The “DealUser” procedure first checks whether or not the user has specified a certain hand or whether he or she has opted to receive a random hand. In the latter case, the cards are shuffled with the “Shuffle” procedure, and the first five cards are given to the user by calling the “Deal” function procedure. If the user has specified his or her own hand, then it has already been dealt, and we need to move those cards to the front of the deck. We then sort the user’s hand with the “SortArray” procedure and display the card images on the simulation sheet with the “GetImage” function procedure.

```
Sub DealUser()  'deal cards to user first
    If RandHand Then
    'if random hand for user, then include their cards in the shuffle
        DealStart = 1
        Call Shuffle
        p = 1
        Call Deal("Player1", 0)
    Else
        'move user's selected cards to front of deck and do not shuffle them
        j = 1
        i = 1
        Do Until j = 6
            If Taken(i) = True Then
                TempSuit = Card(i, 1)
                TempCard = Card(i, 2)
                Card(i, 1) = Card(j, 1)
                Card(i, 2) = Card(j, 2)
                Card(j, 1) = TempSuit
                Card(j, 2) = TempCard
                j = j + 1
            End If
            i = i + 1
        Loop

        p = 1
        Call SortArray
        For i = 1 To 5
            Call GetImage(i)
            ActiveSheet.Paste Destination:=Range("Player1").Offset(1, i - 1)
        Next i
    End If

End Sub
```

Figure CS24.13 The “DealUser” procedure.

The “Shuffle,” “DealOthers,” and “Deal” procedures are illustrated in Figure CS24.14. The first performs the shuffle algorithm described in Section CS4.1.1. “DealOthers” calls the “Deal” function procedure for each player. It deals cards to each player, as described in Section CS4.1.1 and then calls the “SortArray” and “GetImage” procedures.

```
Sub Shuffle()
    'shuffle cards
    For i = DealStart To 52
        j = Int((52 - i + 1) * Rnd() + i)
        TempSuit = Card(i, 1)
        TempCard = Card(i, 2)
        Card(i, 1) = Card(j, 1)
        Card(i, 2) = Card(j, 2)
        Card(j, 1) = TempSuit
        Card(j, 2) = TempCard
    Next i
End Sub

Sub DealOthers()    'deal cards to all other players playing
    For p = 2 To NPlayers
        Call Deal("Player" & p, (p - 1) * 5)
    Next p
End Sub

Function Deal(RangeName, Offset)    'deal a hand of cards
    'since cards are shuffled already, deal by giving top cards of deck = first undealt cards in Card array
    For i = 1 To 5
        PlayerHands(p, i, 1) = Card(i + Offset, 1)
        PlayerHands(p, i, 2) = Card(i + Offset, 2)
    Next i

    'sort the player's hand
    Call SortArray

    If p = 1 Or runs = NRuns Then   'get images of cards dealt
        For i = 1 To 5
            Call GetImage(i)
            ActiveSheet.Paste Destination:=Range(RangeName).Offset(1, i - 1)
        Next i
    End If
End Function
```

Figure CS24.14 The “Shuffle,” “DealOthers,” and “Deal” procedures.

The “SortArray” procedure sorts the cards with the bubble sort algorithm. The “GetImage” procedure copies the images for each card in the player’s hand from the hidden image sheet and pastes them on the simulation sheet. (See Figure CS24.14 for both procedures.)

```
Function SortArray()
    'sort array card values
    For i = 1 To 4
        For j = 1 To 5 - i
            If PlayerHands(p, j, 2) > PlayerHands(p, j + 1, 2) Then
                TempSuit = PlayerHands(p, j + 1, 1)
                TempCard = PlayerHands(p, j + 1, 2)
                PlayerHands(p, j + 1, 1) = PlayerHands(p, j, 1)
                PlayerHands(p, j + 1, 2) = PlayerHands(p, j, 2)
                PlayerHands(p, j, 1) = TempSuit
                PlayerHands(p, j, 2) = TempCard
            End If
        Next j
    Next i
End Function

Function GetImage(i)
    'copy and paste card images
    j = PlayerHands(p, i, 2)

    Select Case PlayerHands(p, i, 1)
        Case "Spade"
            Range("Spades").Offset(1, j - 2).Copy
        Case "Heart"
            Range("Hearts").Offset(1, j - 2).Copy
        Case "Diamond"
            Range("Diamonds").Offset(1, j - 2).Copy
        Case "Club"
            Range("Clubs").Offset(1, j - 2).Copy
    End Select
End Function
```

Figure CS24.15 The “SortArray” and “GetImage” function procedures.

The “CalcChance” procedure finds the status of each player’s hand and compares it with the user’s status. If the user has the highest status compared to all the players, then the user wins and the number of wins for the user is incremented. Each player’s status is determined with the “FindStatus” function procedure.

```
Sub CalcChance()
    'assign numeric values to status based on winning priority, then compare numeric values - highest wins
    StatusVal(1) = "HighCard"
    StatusVal(2) = "OnePair"
    StatusVal(3) = "TwoPair"
    StatusVal(4) = "ThreeOfKind"
    StatusVal(5) = "Straight"
    StatusVal(6) = "Flush"
    StatusVal(7) = "FullHouse"
    StatusVal(8) = "FourOfKind"
    StatusVal(9) = "StraightFlush"

    'find user's status
    p = 1
    Call FindStatus("Player1")
    For i = 1 To 9
        If Status = StatusVal(i) Then
            WinStatus = i
            Winner = "You won!"
            WinCard = HighCard
        End If
    Next i

    'compare with status of other players to see who won
    For p = 2 To NPlayers
        Call FindStatus("Player" & p)
        For i = 1 To 9
            If Status = StatusVal(i) Then
                'see if player's status is equal to user's status
                If i = WinStatus Then
                    'if so, then compare highcards
                    If HighCard > WinCard Then
                        WinStatus = i
                        Winner = "Player " & p & " wins"
                    End If
                'otherwise, if player's status is higher, then player is winner
                ElseIf i > WinStatus Then
                    WinStatus = i
                    Winner = "Player " & p & " wins"
                End If
            End If
        Next i
    Next p

    'if winner never changed from user to another player, then increment user's win count
    If Winner = "You won!" Then
        CountWin = CountWin + 1
    End If
End Sub
```

Figure CS24.16 The “CalcChance” procedure.

The “FindStatus” function procedure finds the status of each player’s hand by checking each status condition sequentially. (See Figures CS4.17, CS4.18, and CS4.19.) First, the number of each suit and each value are counted for the player’s hand. The status is initialized as “High Card,” since this is the lowest ranking hand.

```
Function FindStatus(RangeName)  'determine status of each player's hand
    'initialize count arrays
    For i = 2 To 14
        CountVal(i) = 0
    Next i
    For i = 1 To 4
        CountSuit(i) = 0
    Next i

    'count number of each card and each suit
    For i = 1 To 5
        CountVal(PlayerHands(p, i, 2)) = CountVal(PlayerHands(p, i, 2)) + 1
        Select Case PlayerHands(p, i, 1)
            Case "Spade"
                CountSuit(1) = CountSuit(1) + 1
            Case "Heart"
                CountSuit(2) = CountSuit(2) + 1
            Case "Diamond"
                CountSuit(3) = CountSuit(3) + 1
            Case "Club"
                CountSuit(4) = CountSuit(4) + 1
        End Select
    Next i

    'default status value (lowest value)
    Status = "HighCard"
```

Figure CS24.17 The first part of the “FindStatus” function procedure.

It then checks for “One Pair,” “Two Pairs,” “Three of a Kind,” and “Four of a Kind” referring to the number of cards in each suit and their values. We also check for “Full House” if “One Pair” or “Three of a Kind” was found.

```
    'search for pairs, three of kind, and four of kind
    For i = 2 To 14
        If CountVal(i) = 4 Then
            Status = "FourOfKind"
            HighCard = i
            Exit For
        ElseIf CountVal(i) = 3 Then
            If Status = "OnePair" Then
                Status = "FullHouse"
                If HighCard <> 1 Then
                    HighCard = i
                End If
                Exit For
            Else
                Status = "ThreeOfKind"
                If HighCard <> 1 Then
                    HighCard = i
                End If
            End If
        ElseIf CountVal(i) = 2 Then
            If Status = "ThreeOfKind" Then
                Status = "FullHouse"
                If HighCard <> 1 Then
                    HighCard = i
                End If
                Exit For
            ElseIf Status = "OnePair" Then
                Status = "TwoPair"
                If HighCard <> 1 Then
                    HighCard = i
                End If
            Else
                Status = "OnePair"
                If HighCard <> 1 Then
                    HighCard = i
                End If
            End If
        End If
    Next i
```

Figure CS24.18 The second part of the “FindStatus” function procedure.

If any of these statuses were found, then we do not need to check the other status conditions. However, if the player's status is still "High Card," then we can check for "Flush," "Straight," and "Straight Flush." We then display the player's status and hand on the simulation sheet.

```
    'if none of these found, keep searching
    If Status = "HighCard" Then
        'search for flush
        For i = 1 To 4
            If CountSuit(i) = 5 Then
                Status = "Flush"
                Exit For
            End If
        Next i

        'search for straight
        For i = 1 To 4
            If PlayerHands(p, i + 1, 2) = PlayerHands(p, i, 2) + 1 Then
                Straight = True
            Else
                Straight = False
                Exit For
            End If
        Next i

        'search for straightflush
        If Straight = True Then
            If Status = "Flush" Then
                Status = "StraightFlush"
            Else
                Status = "Straight"
            End If
        End If

        HighCard = PlayerHands(p, 5, 2)
    End If

    'display status and high card value
    Range(RangeName).Offset(0, 1).Value = Status
    Select Case HighCard
        Case 14
            Range(RangeName).Offset(0, 2).Value = "Ace"
        Case 11
            Range(RangeName).Offset(0, 2).Value = "Jack"
        Case 12
            Range(RangeName).Offset(0, 2).Value = "Queen"
        Case 13
            Range(RangeName).Offset(0, 2).Value = "King"
        Case Else
            Range(RangeName).Offset(0, 2).Value = HighCard
    End Select

End Function
```

Figure CS24.19 The third part of the "FindStatus" function procedure.

The only navigational procedure, which applies to the "End" button on the simulation sheet, takes the user back to the welcome sheet.

```
'navigational procedure
Sub EndProg()
    Worksheets("Welcome").Visible = True
    Worksheets("Final Hand").Visible = False
End Sub
```

Figure CS24.20 The navigational procedure.

Summary

Main	Initializes the application and takes the user to the historical data sheet.
ClearPrev	Clears the previous values on all the sheets.
Input form procedures	Records the number of players, the number of runs, and whether the user's hand is random or specified.
Hand selection form procedures	Records the card suits and the values for the user's hand; performs some error checking.
RunSim	Initializes the deck, deals the users, and runs the simulation by shuffling, dealing to other players, and calculating the user's chance of winning.
Initialize	Initializes the card suits and values in the deck.
DealUser	Either deals a random hand or moves the specified cards to the front of the deck.
Shuffle	Shuffles the cards with the "Shuffle" algorithm.
DealOthers	Calls the "Deal" procedure for each player.
Deal	Gives the next five cards in the deck to the player.
SortArray	Sorts each player's hand using the "Bubble Sort" algorithm.
GetImage	Copies and pastes the card images for the player's hand.
CalcChance	Finds each player's status and compares it to the user's status to determine whether or not the user wins.
FindStatus	Considers the status conditions to determine the player's status.
Navigational	Applies to the "End" button on the simulation sheet.

CS24.5 *Re-solve Options*

The user can re-solve this application by selecting either the "Repeat Simulation" or "Deal New Hand" buttons on the simulation sheet. The first button repeats the simulation with the same user hand with the use of the "RepeatSim" procedure. (See Figure CS24.21.) It sets a Boolean variable so that the user's hand is not re-dealt and then recalls the "RunSim" procedure.

```
'resolve option to repeat simulation with same user hand
Sub RepeatSim()
    NoChange = True
    Call RunSim
End Sub
```

Figure CS24.21 The "RepeatSim" procedure.

When the simulation repeats, small changes may occur in the user's chance of winning with the given hand. (See Figure CS24.22.)

The second re-solve option, selected with the "Deal New Hand" button, again shows the input form to the user and allows him or her to change the number of players, the number of runs, or the hand. For example, the user can specify his or her hand so that it is the same but change the number of players; an increase in the number of players may lessen the user's chance of winning (see Figure CS24.23.) Or, the user can keep the same number of players and runs but ask to be dealt a new random hand (see Figure CS24.24.)

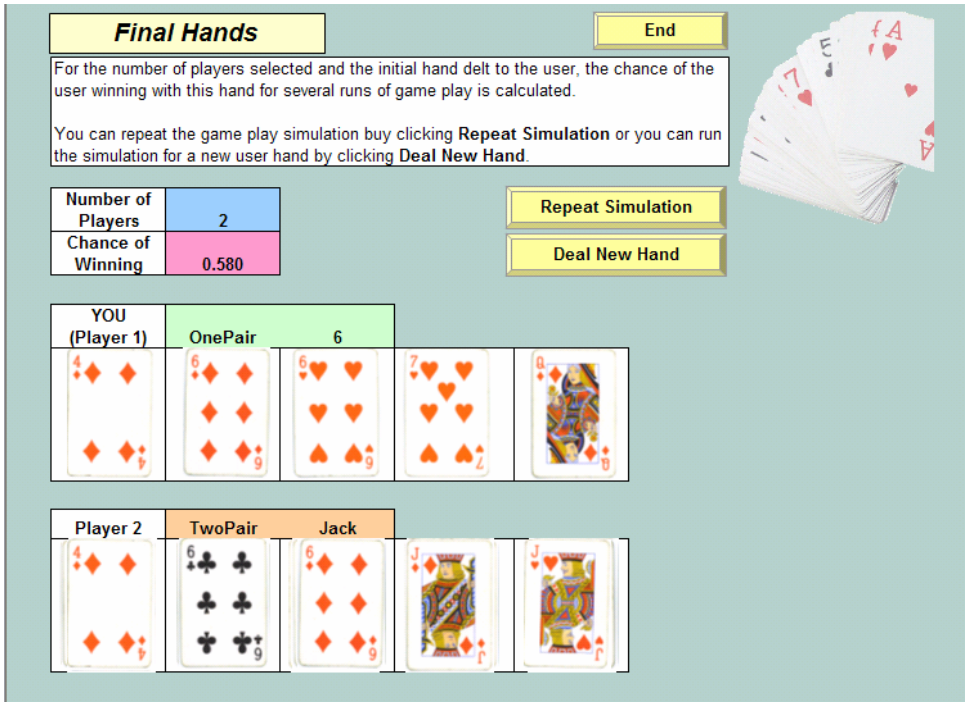

Figure CS24.22 Re-solve option 1: "Repeat Simulation."

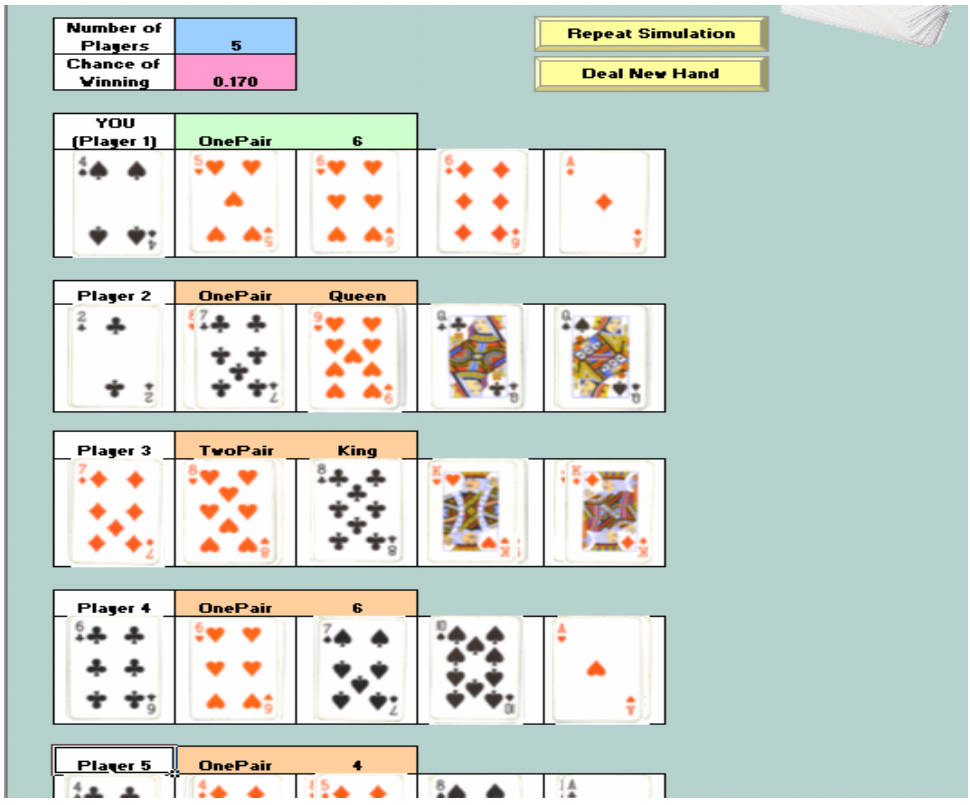

Figure CS24.23 Re-solve option 2: "Deal New Hand," changing the number of players but keeping the same hand.

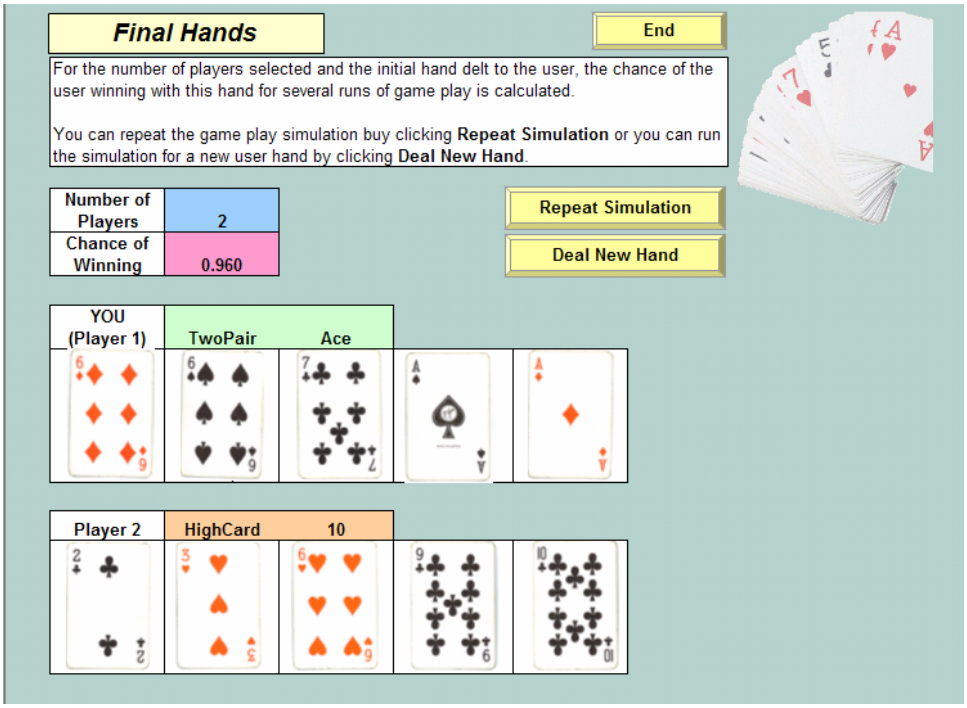

Figure CS24.24 Re-solve option 2: "Deal New Hand," keeping the same number of players, but changing the user's hand.

Summary

Option 1	"Repeat Simulation" button: repeat the simulation with the same user hand.
Option 2	"Deal New Hand" button: specify the same hand with a different number of players or runs; change the user hand with the same number of players; make any other combination of changes using the input form.

CS24.6 *Summary*

- The poker simulation application calculates the chance of winning a game of poker given the hand that the user is dealt and the number of players in the game. The user can either have a random hand dealt or specify his or her hand..
- This application requires three worksheets: the welcome sheet, the simulation sheet, and a hidden image sheet.
- For this application's user interface, we use navigational and functional buttons as well as two user forms.
- Several procedures for this application initialize and perform the simulation for the given user hand and number of players.
- The user can re-solve the application by repeating the simulation for the same user hand, dealing a new user hand, or changing the number of players.

CS24.7 *Extensions*

- Allow the user to specify some cards but not necessarily their entire hand. In other words, suppose that the user wants to know his or her chances of winning if dealt an ace or two kings, but he or she doesn't care about the other cards.
- Allow the user to exchange two or three cards after initially dealt a random hand. Also assume that the other players can switch two or three cards. How does this affect their probabilities for winning?
- Add an option for the user to play poker in "Texas Hold'em" form.
- Create a similar application for playing blackjack.